12-Module

Ex. No. : 12. 1 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Statement:

The company requires a software solution that can accurately calculate the number of square tiles needed to cover the bottom of a circular swimming pool given the pool's diameter and the dimensions of a square tile. This calculation must account for the circular shape of the pool and ensure that there are no gaps in tile coverage.

Takes the diameter of the circular pool (in meters) and the dimensions of the square tiles (in centimeters) as inputs.

Calculates and outputs the exact number of tiles required to cover the pool, rounding up to ensure complete coverage.

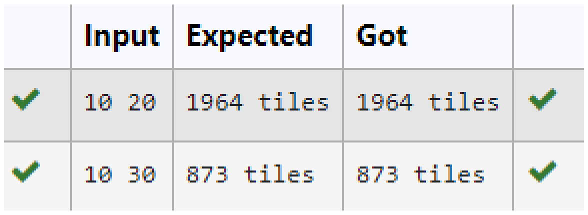
For example:

Input	Result
`10 20`	`1964 tiles`
`10 30`	`873 tiles`

Program:

```
import math


def calculate_tiles(diameter, tile_dimension):

   # Convert diameter from meters to centimeters

   diameter_cm = diameter * 100



   # Calculate the radius of the circular pool
```

```
    radius = diameter_cm / 2


    # Calculate the area of the circular pool's bottom surface
    pool_area = math.pi * (radius ** 2)


    # Calculate the area of a single tile
    tile_area = tile_dimension ** 2


    # Calculate the number of tiles required to cover each square section
    tiles_per_section = math.ceil(tile_area / pool_area)


    # Calculate the total number of tiles required to cover the entire pool area
    total_tiles_required = math.ceil(pool_area / tile_area) * tiles_per_section


    return total_tiles_required

# Main function
if __name__ == "__main__":
    # Input
    while True:
        try:
            diameter, tile_dimension = map(int, input().split())
            # Calculate the number of tiles required

            num_tiles = calculate_tiles(diameter, tile_dimension)


            # Output the result

            print(num_tiles, "tiles")

        except EOFError:

            break
```



	Input	Expected	Got	
✓	10 20	1964 tiles	1964 tiles	✓
✓	10 30	873 tiles	873 tiles	✓

Ex. No. : 12.2 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Statement:

Develop a Python program that manages shoe inventory and processes sales transactions to determine the total revenue generated. The program should handle inputs of shoe sizes available in the shop, track the number of each size, and match these with customer purchase requests. Each transaction should only proceed if the desired shoe size is in stock, and the inventory should update accordingly after each sale.

Input Format:

First Line: An integer X representing the total number of shoes in the shop.

Second Line: A space-separated list of integers representing the shoe sizes in the shop.

Third Line: An integer N representing the number of customer requests.

Next N Lines: Each line contains a pair of space-separated values:

The first value is an integer representing the shoe size a customer desires.

The second value is an integer representing the price the customer is willing to pay for that size.

Output Format:

Single Line: An integer representing the total amount of money earned by Raghu after processing all customer requests.

Constraints:

1≤X≤1000 — Raghu's shop can hold between 1 and 1000 shoes.

Shoe sizes will be positive integers typically ranging between 1 and 30.

1≤N≤1000 — There can be up to 1000 customer requests in a single batch.

The price offered by customers will be a positive integer, typically ranging from $5 to $100 per shoe.

For example:

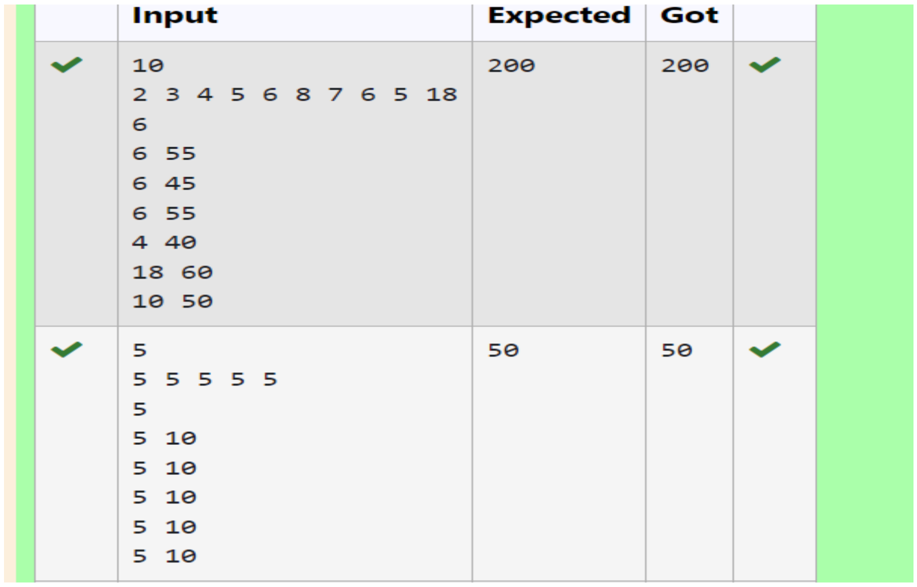
Input	Result

10
2 3 4 5 6 8 7 6 5 18
6
6 55
6 45
6 55
4 40
18 60
10 50
	200

5
5 5 5 5 5
5
5 10
5 10
5 10
5 10
5 10
	50

Program :

```
def calculate_revenue(X, sizes_available, N, requests):
    # Initialize total revenue to 0
    total_revenue = 0


    # Create a dictionary to store the inventory of each shoe size
    inventory = {size: sizes_available.count(size) for size in set(sizes_available)}


    # Process each customer request
    for size, price in requests:
        # Check if the desired shoe size is available in the inventory
        if size in inventory and inventory[size] > 0:
            # Update the inventory after the sale
            inventory[size] -= 1

            # Add the revenue from the sale to the total revenue

            total_revenue += price


    return total_revenue


# Input

X = int(input())  # Total number of shoes

sizes_available = list(map(int, input().split()))  # Available shoe sizes

N = int(input())  # Number of customer requests

requests = [tuple(map(int, input().split())) for _ in range(N)]  # Customer requests


# Calculate and output the total revenue

print(calculate_revenue(X, sizes_available, N, requests))
```



	Input	Expected	Got	
✓	10 2 3 4 5 6 8 7 6 5 18 6 6 55 6 45 6 55 4 40 18 60 10 50	200	200	✓
✓	5 5 5 5 5 5 5 5 10 5 10 5 10 5 10 5 10	50	50	✓

Ex. No. : 12.3 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Problem Statement:

Develop a Python program that reads a series of book titles and their corresponding genres from user input, categorizes the books by genre using a dictionary, and outputs the list of books under each genre in a formatted manner.

Input Format:

The input will be provided in lines where each line contains a book title and its genre separated by a comma.

Input terminates with a blank line.

Output Format:

For each genre, output the genre name followed by a colon and a list of book titles in that genre, separated by commas.

Constraints:

Book titles and genres are strings.

Book titles can vary in length but will not exceed 100 characters.

Genres will not exceed 50 characters.

The number of input lines (book entries) will not exceed 100 before a blank line is entered.

For example:

Input	Result

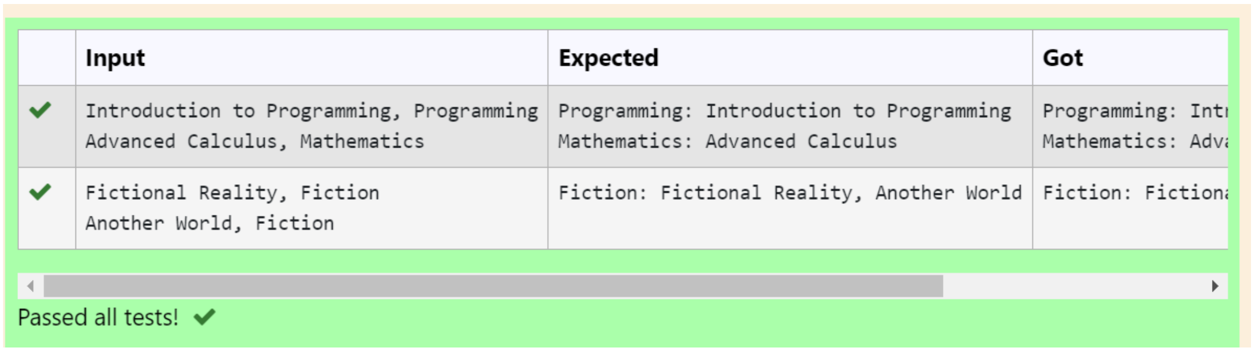
`Introduction to Programming, Programming`
`Advanced Calculus, Mathematics`
	`Programming: Introduction to Programming`
	`Mathematics: Advanced Calculus`

`Fictional Reality, Fiction`
`Another World, Fiction`
	`Fiction: Fictional Reality, Another World`

Program :

```
def gather_input():
    input_lines = []
    while True:
        try:
            line = input().strip()
            if not line:
                break
            input_lines.append(line)
        except EOFError:
            break
    return '\n'.join(input_lines)


def categorize_books(input_lines):
    # List to store tuples of genre and books
    categorized_books = []


    # Read input until a blank line is encountered
    for line in input_lines.split('\n'):
        line = line.strip()
        if not line:
            break
        # Split the input line into book title and genre
        parts = line.split(',')
        if len(parts) != 2:
            continue
book_title, genre = map(str.strip, parts)

        # Check if the genre already exists in the list
        for idx, (existing_genre, _) in enumerate(categorized_books):
            if existing_genre == genre:
                categorized_books[idx][1].append(book_title)
                break
        else:
            # If the genre does not exist, add it to the list
            categorized_books.append((genre, [book_title]))

    return categorized_books

# Main function
if __name__ == "__main__":
    # Gather input from the user

    input_str = gather_input()
```

```
# Categorize the books
categorized_books = categorize_books(input_str)


# Output the books categorized by genre
for genre, books in categorized_books:
    print(f"{genre}: {', '.join(books)}")
```



	Input	Expected	Got
✓	Introduction to Programming, Programming Advanced Calculus, Mathematics	Programming: Introduction to Programming Mathematics: Advanced Calculus	Programming: Intr Mathematics: Adva
✓	Fictional Reality, Fiction Another World, Fiction	Fiction: Fictional Reality, Another World	Fiction: Fictiona

Passed all tests! ✓

Ex. No. : 12.4 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

Given an integer `n`, print `true` *if it is a power of two. Otherwise, print* `false`.

An integer `n` is a power of two, if there exists an integer `x` such that `n == 2`^{x}.

For example:

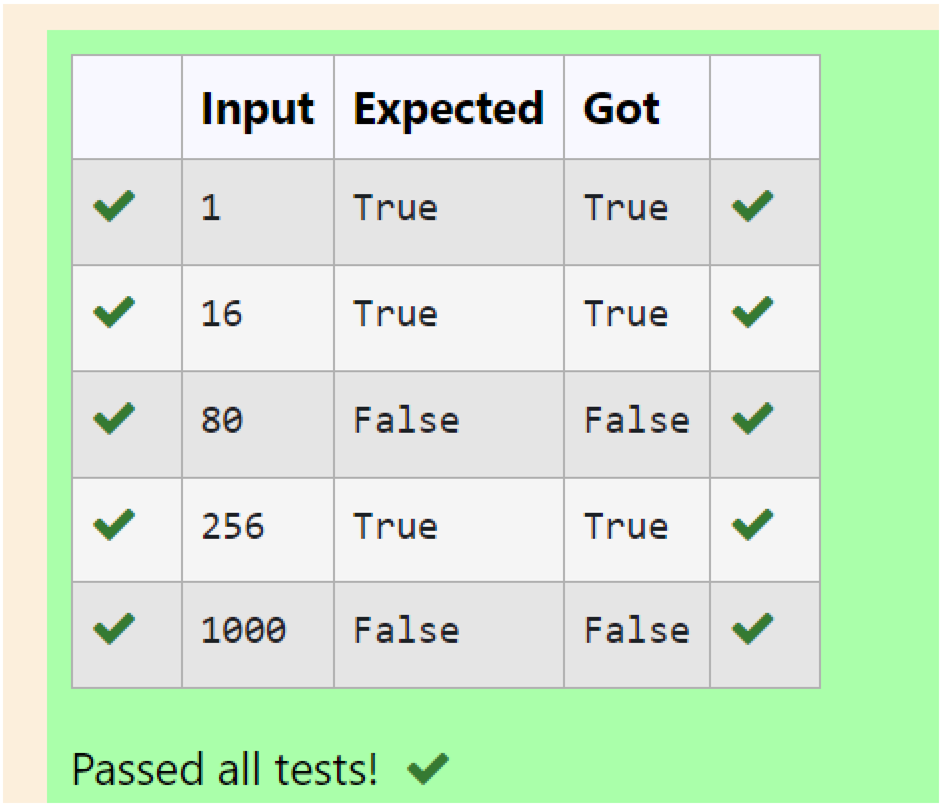
Input	Result
1	True
80	False

Program :

```
def is_power_of_four(n):
    if n <= 0:
        return False

    while n != 1:
        if n % 4 != 0:
            return False
        n //= 4

    return True

# Read input
n = int(input())
```

Check if n is a power of four and print the result

print(is_power_of_four(n))



	Input	Expected	Got	
✓	1	True	True	✓
✓	16	True	True	✓
✓	80	False	False	✓
✓	256	True	True	✓
✓	1000	False	False	✓

Passed all tests! ✓

Ex. No. : 12.5 **Date: 7/06/2024**

Register No.: 231401001 **Name: Aafrin Fathima N**

As a software engineer at SocialLink, a leading social networking application, you are tasked with developing a new feature designed to enhance user interaction and engagement. The company aims to introduce a system where users can form connections based on shared interests and activities. One of the feature's components involves analyzing pairs of users based on the activities they've participated in, specifically looking at the numerical difference in the number of activities each user has participated in.

Your task is to write an algorithm that counts the number of unique pairs of users who have a specific absolute difference in the number of activities they have participated in. This algorithm will serve as the backbone for a larger feature that recommends user connections based on shared participation patterns.

Problem Statement

Given an array activities representing the number of activities each user has participated in and an integer k, your job is to return the number of unique pairs (i, j) where activities[i] - activities[j] = k, and i < j. The absolute difference between the activities should be exactly k.

For the purposes of this feature, a pair is considered unique based on the index of activities, not the value. That is, if there are two users with the same number of activities, they are considered distinct entities.

Input Format

The first line contains an integer, n, the size of the array nums.

The second line contains n space-separated integers, nums[i].

The third line contains an integer, k.

Output Format

Return a single integer representing the number of unique pairs (i, j)

where | nums[i] - nums[j] | = k and i < j.

Constraints:

$1 \leq n \leq 10^5$

$-10^4 \leq nums[i] \leq 10^4$

$0 \leq k \leq 10^4$

For example:

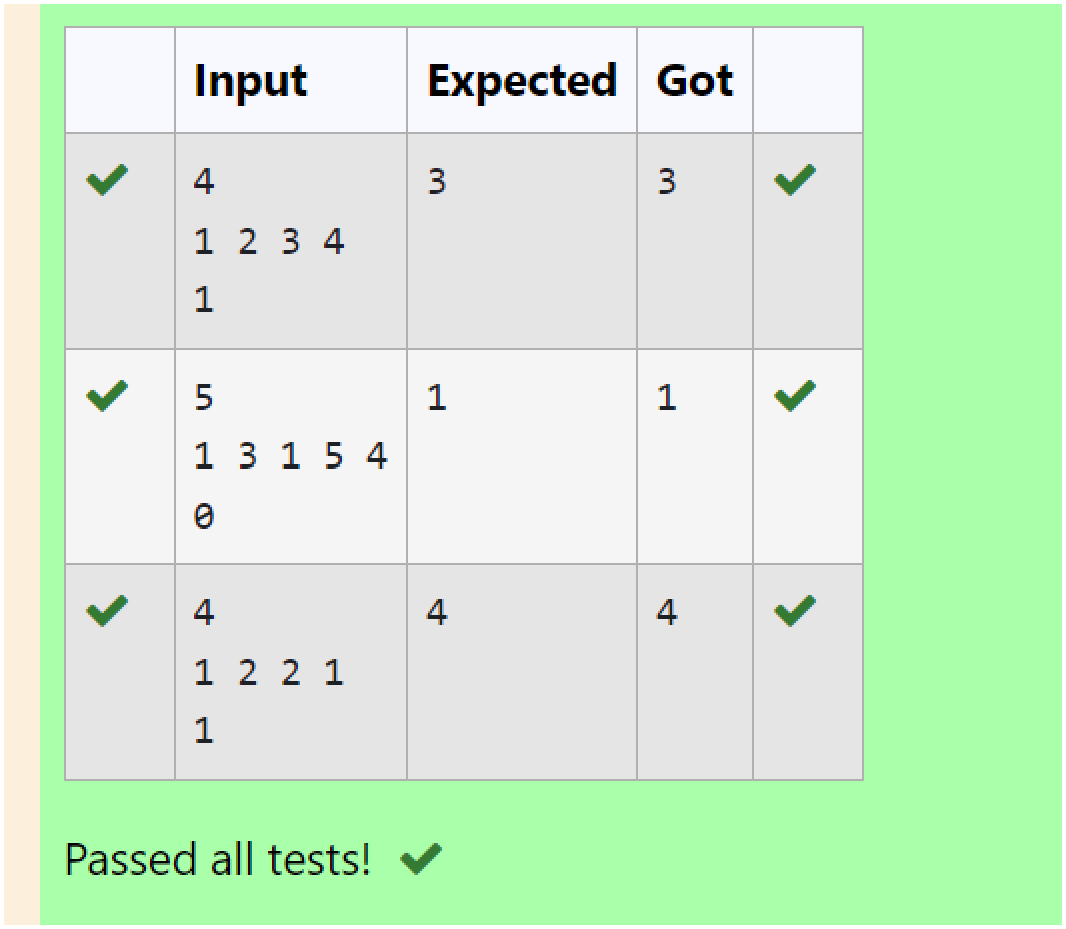
Input	Result

5
1 3 1 5 4
0
	1

4
1 2 2 1
1
	4

Program :

```
def count_unique_pairs(activities, k):

  activity_counts = {}

  unique_pairs = 0

  for activity_count in activities:

    desired_count_pos = activity_count + k

    desired_count_neg = activity_count - k



    # Check for pairs with desired difference

    if desired_count_pos in activity_counts:

      unique_pairs += activity_counts[desired_count_pos]

    if desired_count_neg != activity_count and desired_count_neg in activity_counts:  #
Avoid double counting

      unique_pairs += activity_counts[desired_count_neg]
    # Update activity count in hash table
    activity_counts[activity_count] = activity_counts.get(activity_count, 0) + 1
  return unique_pairs
n = int(input())
activities = list(map(int, input().split()))
k = int(input())
print(count_unique_pairs(activities, k))
```



	Input	Expected	Got	
✓	4 1 2 3 4 1	3	3	✓
✓	5 1 3 1 5 4 0	1	1	✓
✓	4 1 2 2 1 1	4	4	✓

Passed all tests! ✓